APPENDIX

APPENDIX

1. Computer Operation and Related Forms of Notation

The modern digital computer is a highly versatile device that can be employed to manipulate data provided to it in a variety of different ways. The data may represent names, numbers, addresses, dictionary words, or any other information. Normally the data is stored and organized in the computer in the form of electrical signals and the electrical signals are manipulated in the computer to perform various functions such as organizing names and addresses in a particular order, effecting computations with the numbers, or achieving a translation from one language to another.

Computers have a number of distinct features including a potential for vast data capacity, extremely high speed of operation and—most pertinent here—"stored program control." Stored program control is a technique used since the mid-1940's in which a sequence of electrical control signals is stored in a retrievable form within the computer for the purpose of controlling the operation of the computer itself to perform a particular function. These control signals, in contradistinction to signals representing the data to be processed, effect two basic types of control: (1) they effect the transfer of signals to be processed from one part of the machine to another, and (2) they effectively reconnect, by opening and closing electronic switches, the various hardware components of the machine so that a wide variety of operations can be performed with a common set of apparatus components.

A series of control instructions is commonly known as a "program." Each step in this series may be described in ordinary English text (*e.g.*, "shift the data in register D to the right three places in the register"). It is then written in one of several "computer languages" (*e.g.*, "SHR 3"). By means not here relevant the program

is ultimately converted into a sequence of electrical signals and stored within the computer. The program thus identifies a set of control steps which can be performed in the computer in the pre-determined pattern. The data to be processed is, of course, also introduced and processed as electrical signals.

The use of stored program control is not restricted to machines whose primary purpose is to perform arithmetic computations. It is also used to control chemical plants, machine tools and telephone switching systems such as that of respondents. These special purpose applications may use apparatus elements or "hardware" which differ substantially yet all involve the manipulation of data signals under the direction of stored program control signals in the manner just described.

Although the computer can be used to obtain results that could also be achieved independently by an individual in his mind or with a pencil and paper, it is a misnomer to regard the computer as if it "thinks" in any human sense. The computer is a machine in which electrical signals are manipulated in accordance with programmed steps predesignated by an individual. If the most efficient use is to be made of the computer, then the programmed steps normally differ substantially both in substance and in sequence from the steps an individual would follow if seeking to obtain the same result with a pencil and paper.

A digital computer stores and utilizes all of its data in the form of discrete electrical signals. Thus a discrete signal or sequence of such signals may represent a person's name, instructions, numbers, or any other data. The work stuff of the computer, therefore, is sequences of electrical signals which, in turn, represent the basic data the user wishes to organize, store and retrieve, or otherwise process.

If the electrical signals are to represent numbers, the numbers could be represented in decimal form; for exam-

ple, each decimal digit could be represented in a computer by one of ten levels of voltage in an electrical device (*e.g.*, a transistor). Most computers are designed, however, to record and utilize numbers in binary form wherein all numbers are expressed by sequences of the digits 1 and 0 rather than by the conventional 10 digits of the decimal system (*i.e.*, 0 through 9). Among other reasons for the use of binary numbers is the fact that it is more efficient to store and utilize electrical signals in a computer where this can be done with only two states, such as the "on" and "off" states of an electrical device rather than 10 states.

In the binary system digits have different "positional values" than they have in the decimal system. The positional value is the numerical value that a digit has because of its location in a particular position in a number. Thus in the decimal system, the positional values of the digits in the number 111, reading from right to left, are respectively 1, 10, and 100, *i.e.*, ascending powers of the "base number" or "radix" 10. In the binary system, by contrast, each digit is assigned positional values based on ascending powers of the base number 2, increasing from right to left as follows:

128 64 32 16 8 4 2 1

In binary notation, the decimal number 1 is represented by placing a 1 in the first position and a 0 in all other positions as shown in the following table which, for convenience, contains the positional values above each digit:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	0	1

The decimal number 3 is written in binary notation by placing a binary 1 in the first and second positions as follows:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1

Similarly, the decimal number 53 is written in binary notation by placing a binary 1 in the first, third, fifth and sixth positions as follows:

128	64	32	16	8	4	2	1
0	0	1	1	0	1	0	1

The data the programmer wishes to store in the computer is often most conveniently available in the form of a decimal number rather than a binary number. For instance, a series of social security numbers may be intended for storage in a computer memory; similarly, a standard telephone number employs decimal digits. Since the normal digital computer stores and utilizes the data in the form of binary signals (see p. 3a, above), the decimal digits of the social security or telephone number must be converted into binary form.

This conversion is often simplified, for reasons explained below, by employing a two step process in which the number is first converted from decimal to another number form called binary-coded-decimal or BCD form and then converted from BCD form to binary form. In the BCD system, as in the binary system, only the digits 1 and 0 are utilized. The positional values of the digits, however, are different. Thus, a given sequence of 1's and 0's will represent one numerical value if it is read as a BCD number and a different numerical value if it is read as a binary number.

To convert a decimal number to BCD form, each digit of the decimal number is separately converted to its binary equivalent expressed as a set of four binary digits; the sets of binary digits placed in sequence *are* the BCD number. Thus, the decimal number 53 is represented in BCD notation by combining the binary equivalents of the decimal digits 5 (0101) and 3 (0011) into the BCD number 0101/0011. This sequence is not a true binary number because the positional values of the digits differ from pure binary notation (where

the decimal number 53 would be expressed as 110101). The positional values of the first eight BCD digits are:

80 40 20 10 / 8 4 2 1

It will be observed that the first four positions from the right are assigned the same values as in binary notation, but that the next four positions (and all positions thereafter) have different values. Any number less than ten is therefore written the same in both the binary and the binary-coded-decimal notation. For example, the number three is the same in both notations.

80	40	20	10	8	4	2	1
0	0	0	0	0	0	1	1

However, the BCD number ten is different from the binary number ten (1010). In BCD notation, ten is represented by:

80	40	20	10	8	4	2	1
0	0	0	1	0	0	0	0

As noted above, the BCD number fifty-three is written:

80	40	20	10	8	4	2	1
0	1	0	1	0	0	1	1

The reason for converting decimal numbers to BCD form before conversion into binary numbers arises from the fact that it is often convenient or necessary to convert separately each decimal digit one at a time. For example, in a telephone each turn of the dial transmits a set of pulses representing one decimal digit. If each set as it arrives at the control apparatus is converted into a group of four binary signals equivalent to the original digit, then the entire sequence of signals after all the pulses have been converted represents the original telephone number in BCD form. The second step of the conversion, from BCD to binary form, can then be effected in a number of different ways. It would require more elaborate circuitry to store each set of pulses as it arrived and then to convert all of the sets directly into the final binary sequence.

II. Respondents' Invention

At the time respondents made the invention involved in this case, there were a number of different known ways of converting electrical signals representing BCD numbers into signals representing equivalent binary numbers. One important method of conversion prior to respondents' invention involved the storage in the computer of a range of conversion values. A conversion value for purposes of BCD-to-binary-conversion is the binary number representing the positional value of a given BCD digit. Thus, the conversion value for the fifth digit in a BCD number whose positional value is 10 is the numerical value 10 expressed as a binary number, *i.e.*, 1010.

Using stored conversion values the computer would be programmed to retrieve from storage the signals representing the conversion values for the successive digits in the BCD number to be converted; for each position of the BCD number (after the first four) in which a 1 rather than a 0 appeared, the appropriate conversion value would be added to the first four digits (which are already in pure binary form). The sum of these conversion values would be taken by the steps of binary addition, and the resulting number would be the pure binary equivalent of the original BCD number.

This method required that the computer be equipped for multi-digit addition and also utilized substantial storage space in the computer's electrical memory. Both of these requirements presented substantial difficulties for respondents because the PBX telephone system they were developing lacked a multi-digit adding device and had only limited storage capacity. A further disadvantage of the existing method was that it tended to require the movement of many different signals representing different conversion values and each movement increased the possibility of error through circuit failure.

A number of other computer conversion techniques were also available, one of the most significant being the technique claimed in the Couleur patent which is described in

the brief of amicus Mobil (p. A11). Among other difficulties the Couleur technique required the computer to possess a circuit for subtracting electrical pulses representing the numerical value 3 and a circuit for detecting electrical pulses representing numerical values of 8 or more. Respondents' PBX possessed neither of these circuits.

Based upon the discovery of a mathematical relationship which is described below (pp. 14a-20a) respondents invented a new, useful and unobvious way of converting signals representing BCD numbers into signals representing equivalent binary numbers. The process required neither a multi-digit adder nor a large amount of computer storage capacity. Furthermore, it did not need either of the circuits utilized in the technique covered by the Couleur patent. It permitted conversion to be made with the relatively simple mechanisms available in the PBX and it reduced the possibility of error from circuit failure by reducing the number of necessary signal communications within the PBX.

In essence, the invented process accomplished conversion of the signals representing the BCD number by subjecting the signals to a series of repetitive steps in a particular pattern. In each complete cycle, the computer would "test" (identify) the signal at a position in the register (which is a device to contain the BCD signals to be converted); if the test identified a pulse (representing the digit 1) it would "mask" (erase) the pulse; and, in that event, it would "add" (increment) at each of two register positions predesignated in the sequence of signals representing the BCD number.

Through the use of a reentrant shift register, the set of signals being converted would in fact be shifted past the testing, masking and incrementing circuits rather than having separate circuits to perform each function at each separate position of the register containing the sequence of signals. The reentrant shift register is merely a type of register that allows a sequence of signals in the register to be shifted to the left or right in such a way that signals

exiting at one end of the register reappear in correct order at the other end of the register. See diagrams at pp. 9a-10a, below. The use of the register further facilitated the conversion process invented by the respondents.

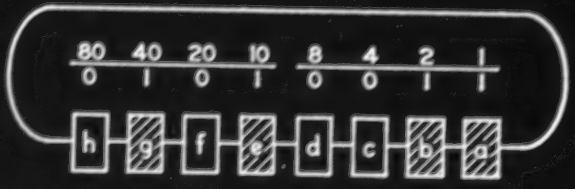
When a sequence of signals representing a BCD number is stored in a register for conversion, the signals in the first four register positions at the right need not be converted since the BCD and binary notations are identical in the first four positions. To convert the signals in the next four positions, the sequence of steps involving testing, shifting, masking and incrementing is performed on the signals initially located in the fifth through eighth register positions, one signal at a time. After these steps have been performed, those signals have been converted into a sequence of signals representing the equivalent binary number and have been cumulated with the existing signals originally stored in the first four positions of the register. This entire sequence of signals represents a binary number equivalent in value to the BCD number previously represented in the register.

The operation of the process can be graphically represented by diagramming the reentrant shift register and following the process through each of the steps necessary to convert a set of eight signals from BCD form into binary form. In the diagrams pictured below, the testing and masking operations are always performed at the second ("b") position and the incrementing is always performed at the first ("a") position. These circuits can therefore be permanently fixed to these registers.

The reentrant shift register can be pictured as follows:

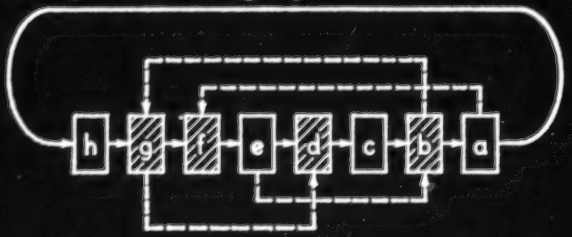
The eight boxes, a through h, are called digit registers and are each capable of holding one binary signal, usually represented by the presence or absence of a particular kind of electrical signal.

The reentrant shift register can be used to store binary or BCD numbers simply by storing the appropriate electrical signals in the digit registers. An example is the BCD number representing 53, as shown below, with the presence of a signal indicated by cross-hatching in the appropriate boxes. Storing these BCD signals in the reentrant shift register corresponds to step (1) of claim 8 of respondents' patent application ("storing the binary-coded-decimal signals in a reentrant shift register"):



The reentrant shift register shifts electrical signals to the right or to the left by moving each signal condition to the digit register to the right or left, respectively, of its starting position. Signals shifted out of one end of the reentrant shift register are inserted into (reenter) the other end of the register. For example, the second step of respondents' claim 8 calls for "shifting the signals to the right by at least three places, until there is a binary '1' in the second position of said register." This step requires a test at register b to determine if a binary 1 is actually present after the shift has occurred. If no binary 1 is present, shifting continues until a binary 1 is present in register

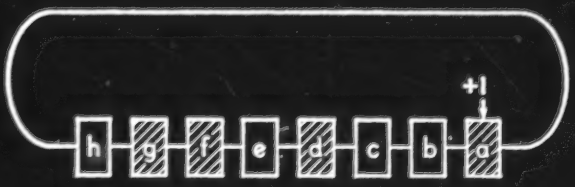
b. In this example, only three shifts are required. When this is done, the following signal conditions exist:



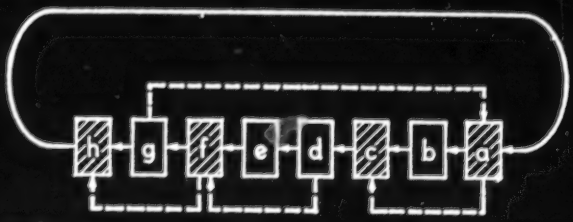
The dotted arrows indicate the movement of each signal. Thus, the signal in register a was moved three places to the right (out of the right end and into the left end) to register f; the signal at register b was moved three places to the right (also out the right end) to register g; the signal in register e was moved three places to the right to register b; and the signal in register g was moved three places to the right to register d.

The third step of claim 8 calls for "masking out said binary '1' in the second position [b] of said register."

The fourth step of claim 8 requires "adding a binary '1' to the first position [a] of said register" as follows:

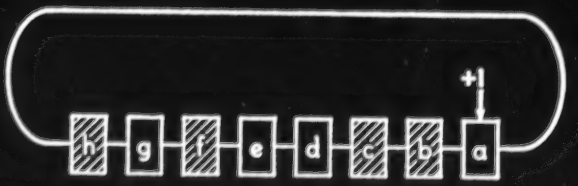


The fifth step of claim 8 calls for "shifting the signals to the left by two positions" as follows:



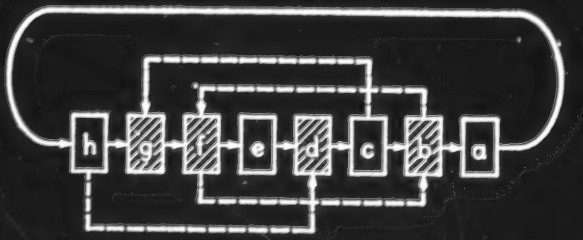
The dotted arrows represent the actual movement of the signals.

The sixth step of claim 8 calls for "adding a binary '1' to said first [a] position" of the register:



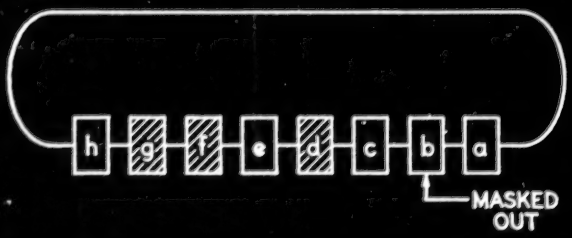
Register a is not shaded because the addition of a binary 1 to that register, which already contains a binary 1, causes a carry to register b which is therefore shown shaded.

The seventh step of claim 8 calls for "shifting the signals to the right by at least three positions in preparation for a succeeding binary '1' in the second [b] position of said register." In our example, the signals must be shifted to the right *four* positions to bring the succeeding binary 1 to the second position of the register:



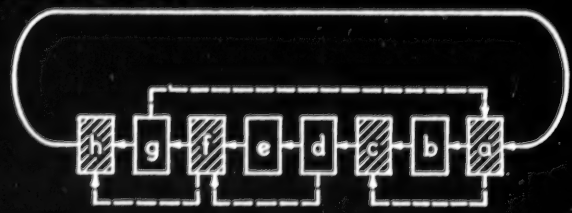
The dotted arrows again indicate the overall movement of each signal.

The conversion is completed by repeating from step (3) onward. This requires masking out the signal in the second (b) position:

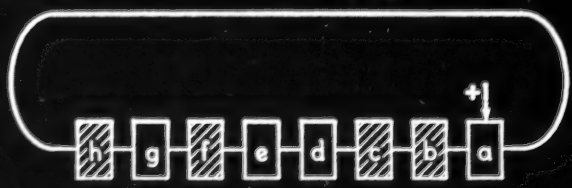


and then adding a binary 1 in the first (a) position:

The signals in the shift register are then shifted to the left by two positions:

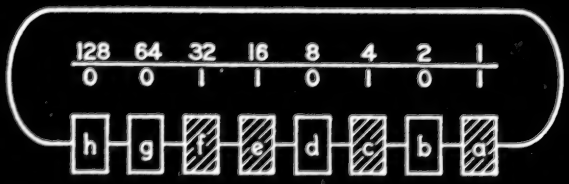


and a binary 1 is added to the first (a) position:



Again, register a is not shaded and register b is shaded due to the carry from the addition.

As a final step, the signals are shifted to the right by five places (or to the left by three places) to produce the final result:



which represents 53 in binary notation.

The register may, of course, contain more than eight positions and the BCD number may have a numerical value that requires more than eight signals. The repetitive steps of respondents' invented process suffice ultimately to convert signals representing a BCD number to signals representing an equivalent binary number no matter how many BCD signals are initially stored in the register. See pp. 17a-20a, below.

III. The Mathematical Discovery

Respondents were assisted in developing their machine process invention by a new mathematical discovery they made in the course of their efforts to solve the problem of converting BCD into binary signals in their PBX control unit. This discovery involved the ascertainment of certain relationships between given BCD numbers, their binary equivalents, and the steps for converting one into the other; and from these relationships, the respondents derived a new way of converting any BCD number into its binary equivalent.

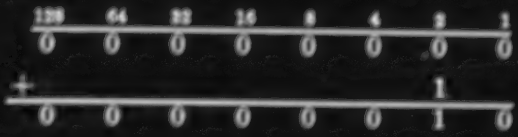
When respondents made their discovery, there already existed a number of different mathematical techniques for converting BCD numbers to binary numbers. A familiar

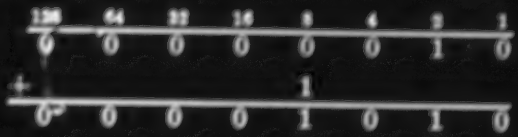
way of converting by hand is described in the petitioner's brief (p. 12, n.14). In this method, each four-digit BCD number (after the first four digits which need no conversion)—corresponding respectively to the original tens position decimal digit, the hundreds position decimal digit, and so on—is converted in turn from a BCD number into an equivalent binary number by the rules of binary multiplication. The sum of the first four digits and of all of the converted numbers, now in binary form, is taken by the rules of binary addition.

Thus, to convert the BCD number representing the decimal number 53, the BCD number in the tens position, *i.e.*, the BCD number representing the decimal digit 5, would be converted into the pure binary number representing the numerical value 50 through binary multiplication of the BCD number representing 5 (0101) by the binary number representing its positional value, *i.e.*, 10 stated in binary form (1010). The resulting binary number would be added to the existing number representing the decimal digit 3 (which is the same in binary and BCD form) to produce the result. This technique can be applied, in the manner described above, to BCD numbers of any length.

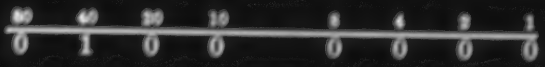
In translating BCD numbers to equivalent binary numbers, it has already been noted that the BCD and binary expressions are identical in the case of all numerical values of 9 or less. See p. 5a, above. The problem of conversion, therefore, begins with BCD numbers representing numerical values of 10 or more. What respondents discovered, in deriving their novel technique, was that all BCD numbers could be converted into binary numbers of equivalent values by the addition of binary 1's in a particular sequence at designated positions.

The individual digits of the tens positions of a BCD number can be separately converted to binary notation in the following manner. A 1 in the BCD 10's position:

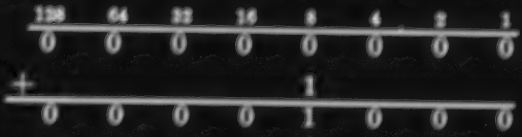
can be converted to binary notation by adding a 1 at the 2's position:

and then adding a 1 in the 8's position.



A 1 in the BCD 20's, 40's and 80's positions can be similarly converted. For example, a 1 in the 40's position:



can be converted to binary notation by adding a 1 at the 8's position:



and then adding a 1 in the 32's position:

In considering such conversions, respondents first noted that it was possible to convert *all* BCD numbers using repetitive additions of binary 1's. Respondents also noted that the additions of 1's for all conversions took place at regularly spaced binary positions removed by a fixed number of positions from the 1's being converted. This discovery, made by respondents and not available in the prior art, provided a direct relationship between any BCD number and the equivalent binary number.

As an example, the BCD number having the value 53:

can be converted to binary notation by first noting that the units value 3 needs no conversion:

The BCD value of 50:

can be converted as noted above. First, the 1 in the position having a BCD value of 10 is converted by adding a binary 1 to the 2's position as follows:

and then adding a binary 1 to the 8's position:

Second, the 1 in the position having the BCD value of 40 is then converted by adding a binary 1 to the 8's position as follows:

128	64	32	16	8	4	2	1
0	0	0	0	1	1	0	1
+				1			
0	0	0	1	0	1	0	1

and then adding a 1 to the 32's position:

128	64	32	16	8	4	2	1
0	0	0	1	0	1	0	1
+		1					
0	0	1	1	0	1	0	1

to complete the conversion.

Any other BCD number can be converted in a similar manner. To convert a BCD number having hundreds positions, it is necessary to first convert the hundreds to tens, and then to convert the augmented tens positions to units positions.

As an example, the BCD number representing 253:

800	400	200	100	80	40	20	10	8	4	2	1
0	0	1	0	0	1	0	1	0	0	1	1

can be converted to binary notation by noting that the 1 in the 200's position must first be converted to the tens positions and added to the pre-existing value of 5 in the tens positions. This is accomplished with steps similar to those noted above as follows.

It should be first noted that the BCD value of 50 is expressed in the tens positions as follows:

1280	640	320	160	80	40	20	10
0	0	0	0	0	1	0	1

(The extra four positions (160, 320, 640, 1280) are provided to permit the tens positions to be augmented or increased as the conversion process proceeds.)

The 1 in the position having a BCD value of 200 is converted by first adding a binary 1 to the 40's position as follows:

1280	640	320	160	80	40	20	10
0	0	0	0	0	1	0	1
+					1		
0	0	0	0	1	0	0	1

(The units positions of the BCD number have not been shown in this first stage of the conversion process.)

The conversion of the 1 in the BCD position having a value of 200 is then completed by adding a binary 1 to the 160's position:

1280	640	320	160	80	40	20	10
0	0	0	0	1	0	0	1
+			1				
0	0	0	1	1	0	0	1

Since there are no more binary 1's in the hundreds positions of the BCD number, the conversion to tens is complete. (If there were more binary 1's in the hundreds positions, *e.g.*, a 1 in the 800's position, they would be converted in a similar manner, *e.g.*, converting the 1 in the 800's position by adding binary 1's to the 160's position and the 640's position.

The next step in the conversion process is to convert the now augmented tens positions:

1280	640	320	160	80	40	20	10
0	0	0	1	1	0	0	1

to units. The original units value 3 needs no conversion:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1

The augmented tens positions can be converted to units in exactly the same manner as was used for unaugmented tens

positions. First, the 1 in the augmented tens position having a value of 10 is converted by adding a binary 1 to the 2's position as follows:

128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1
+						1	
0	0	0	0	0	1	0	1

and then adding a binary 1 to the 8's position:

128	64	32	16	8	4	2	1
0	0	0	0	0	1	0	1
+				1			
0	0	0	0	1	1	0	1

The 1 in the augmented tens position having a value of 80 is converted by adding a binary 1 to the 16's position:

128	64	32	16	8	4	2	1
0	0	0	0	1	1	0	1
+			1				
0	0	0	1	1	1	0	1

and then adding a 1 to the 64's position:

128	64	32	16	8	4	2	1
0	0	0	1	1	1	0	1
+	1						
0	1	0	1	1	1	0	1

Finally, the 1 in the augmented tens position having a value of 160 is converted by adding a binary 1 to the 32's position:

128	64	32	16	8	4	2	1
0	1	0	1	1	1	0	1
+		1					
0	1	1	1	1	1	0	1

and then adding a binary 1 to the 128's position:

128	64	32	16	8	4	2	1
0	1	1	1	1	1	0	1
+ 1							
1	1	1	1	1	1	0	1

to complete the conversion.

BCD numbers with even higher valued digit positions (1000's, 10,000's, etc.) can be converted by beginning at the highest valued positions and converting them to the next highest valued positions, then converting the augmented next highest valued positions to the third highest valued positions, and so forth. The conversion is complete when all higher valued positions have thus been converted to units.